What's new with GitOps and OpenShift

Jonas Janz
AppDev Solutions Architect

https://github.com/pixeljonas

What is GitOps?

GitOps is when the infrastructure and/or application state is fully represented by the contents of a git repository. Any changes to the git repository are reflected in the corresponding state of the associated infrastructure and applications through automation.

GitOps is a natural evolution of Agile and DevOps methodologies

Why GitOps?

It takes weeks (or months!) to provision an environment

The application behaves different in production than it did in test

Environments are all manually configured ("pets vs. cattle")

Production deployments have a very low success rate

I have no visibility or record of configuration changes in environments

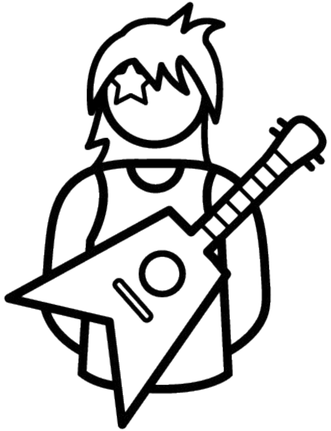
I can't easily rollback changes to a specific version

I can't audit configuration changes

GitOps Benefits

- All changes are auditable
- Standard roll-forward or backwards in the event of failure
- Disaster recovery is “reapply the current state of the manifests”
- Experience is “pushes and pull-requests”

GitOps is for Everyone



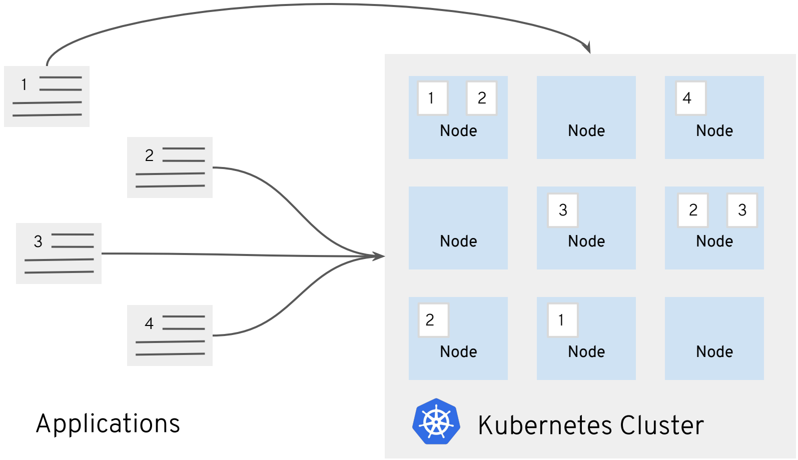
Developers

Operations

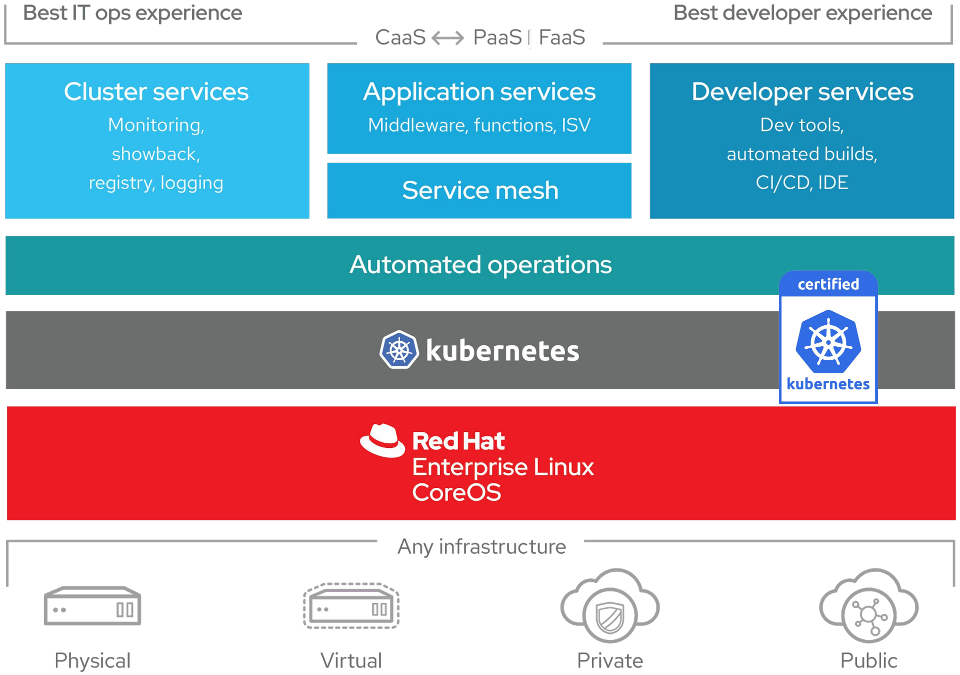
KUBERNETES 101

Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications.

TLDR; It is a resource scheduler



OpenShift 4 - A Smarter Kubernetes Platform



Automated, full-stack installation from the container host to application services

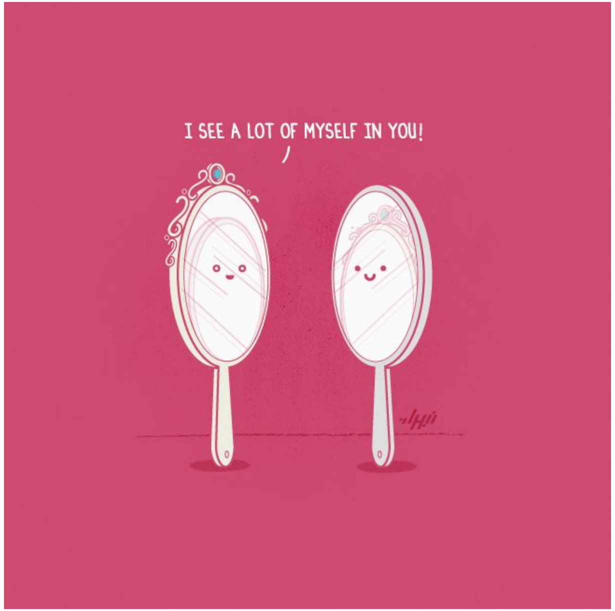
Seamless Kubernetes deployment to any cloud or on-premises environment

Autoscaling of cloud resources

One-click updates for platform, services, and applications

OpenShift and GitOps - A Perfect Match

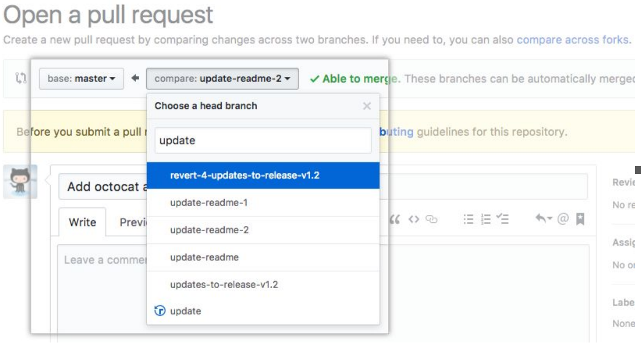
- OpenShift is a declarative environment
  - Cluster configuration is declared and Operators make it happen
  - Application deployments are declared and Kubernetes scheduler makes it happen
- GitOps in traditional environments requires automation/scripting, declarative environment minimizes or eliminates this need
- Declarations are yaml files which are easily stored and managed in git



OpenShift GitOps Principles

- **Separate** application source code (Java/.Net/etc) from manifests (yaml)
- Deployment manifests are standard k8s manifests
- Avoid **duplication** of yaml across environments
- Manifests should be applied with **standard** Openshift and k8s tooling

Day 2 operations : All changes triggered from Git



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base: master ▾ ← compare: update-readme-2 ▾ ✓ Able to merge. These branches can be automatically merged

Choose a head branch ×

update

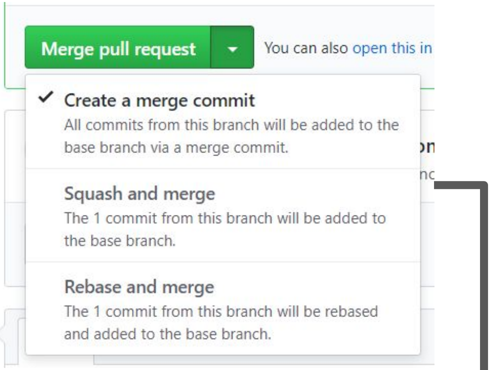
- revert-4-updates-to-release-v1.2
- update-readme-1
- update-readme-2
- update-readme
- updates-to-release-v1.2
- update

Before you submit a pull r... buting guidelines for this repository.

Add octocat a

Write Previ

Leave a commen

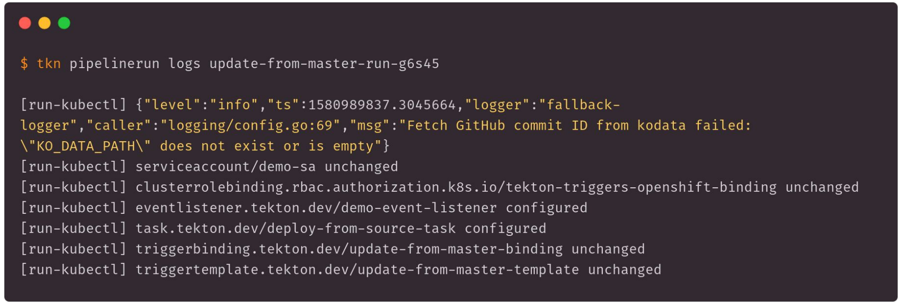


Merge pull request ▾ You can also open this in

✓ Create a merge commit
All commits from this branch will be added to the base branch via a merge commit.

Squash and merge
The 1 commit from this branch will be added to the base branch.

Rebase and merge
The 1 commit from this branch will be rebased and added to the base branch.



```
$ tkn pipelinerun logs update-from-master-run-g6s45

[run-kubectl] {"level":"info","ts":1580989837.3045664,"logger":"fallback-
logger","caller":"logging/config.go:69","msg":"Fetch GitHub commit ID from kodata failed:
\"KO_DATA_PATH\" does not exist or is empty"}
[run-kubectl] serviceaccount/demo-sa unchanged
[run-kubectl] clusterrolebinding.rbac.authorization.k8s.io/tekton-triggers-openshift-binding unchanged
[run-kubectl] eventlistener.tekton.dev/demo-event-listener configured
[run-kubectl] task.tekton.dev/deploy-from-source-task configured
[run-kubectl] triggerbinding.tekton.dev/update-from-master-binding unchanged
[run-kubectl] triggertemplate.tekton.dev/update-from-master-template unchanged
```

Tools of the Trade

Argo CD

Kustomize

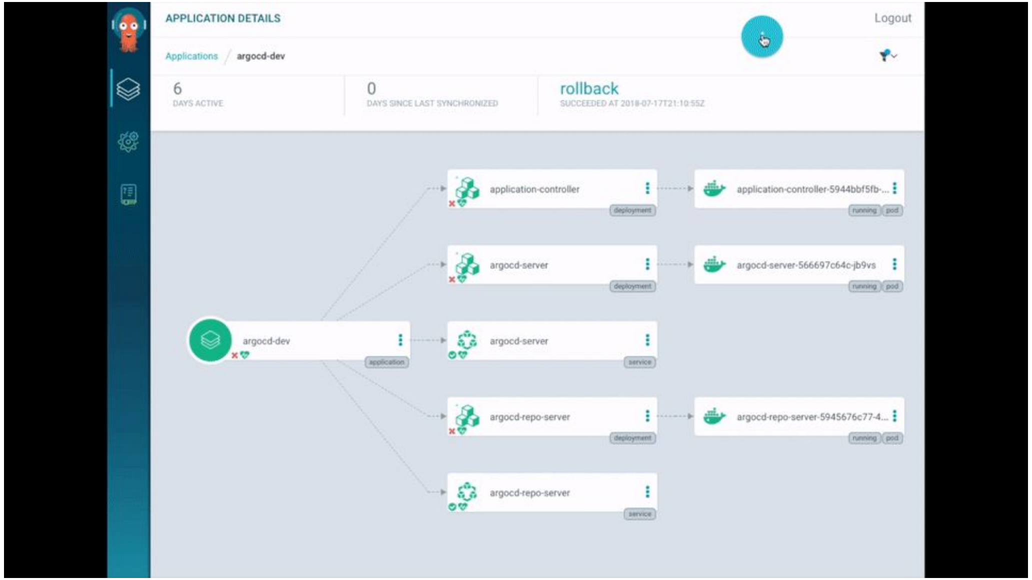
https://argoproj.github.io/argo-cd

https://kustomize.io/

Argo CD - What is It?

Argo CD is a declarative, GitOps continuous delivery tool for Kubernetes.

- Easily deploy applications in a declarative way
- Synchronizes cluster state with git repos
- Works with a variety of Kubernetes deployment tools including:
  - Helm
  - Kustomize
  - Ksonnet/Jsonnet
  - Directories of yaml
- **It is not a CI tool**



APPLICATION DETAILS

Applications / argocd-dev

6 DAYS ACTIVE

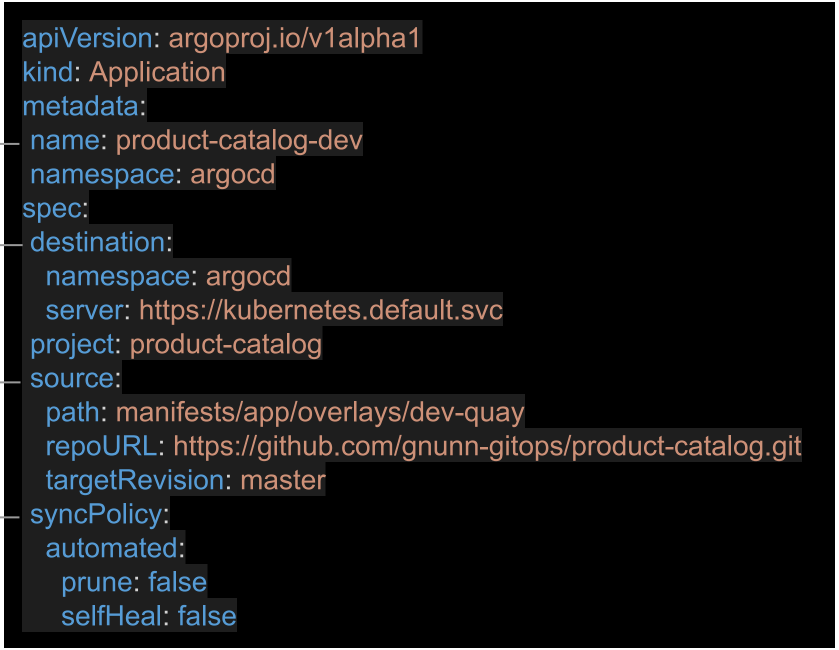
0 DAYS SINCE LAST SYNCHRONIZED

rollback SUCCEEDED AT 2018-07-17T21:10:55Z

Red Hat

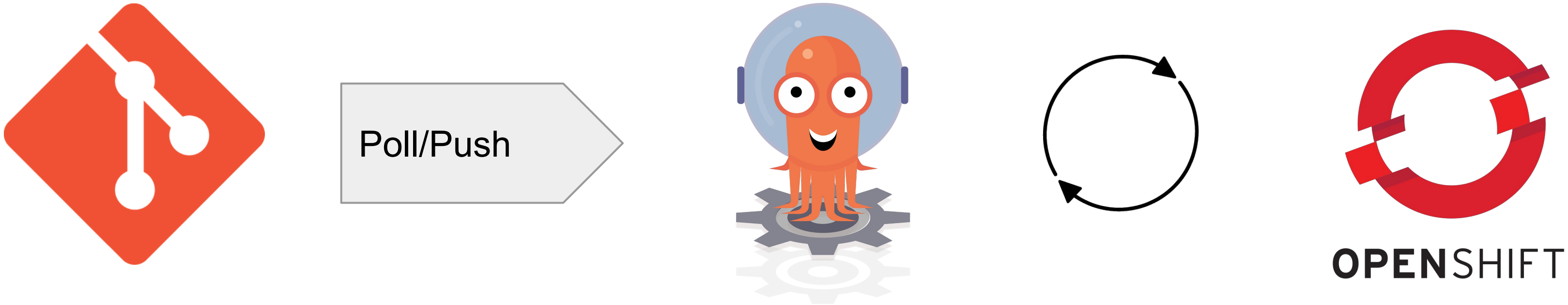
What is an Argo CD Application?

- Argo CD Application is a Custom Resource (CR) that defines the app in a declarative manner
- Application definition includes:
  - Name
  - Cluster
  - Git repository
  - Synchronization Policy
- Applications can be deployed from Argo CD GUI or CLI (argocd or kubectl or oc)



```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: product-catalog-dev
  namespace: argocd
spec:
  destination:
    namespace: argocd
    server: https://kubernetes.default.svc
  project: product-catalog
  source:
    path: manifests/app/overlays/dev-quay
    repoURL: https://github.com/gnunn-gitops/product-catalog.git
    targetRevision: master
  syncPolicy:
    automated:
      prune: false
      selfHeal: false
```

Argo CD - Synchronizing



Poll/Push

Change in git

Pushed to Argo CD

Check
Sync Status

Synchronize

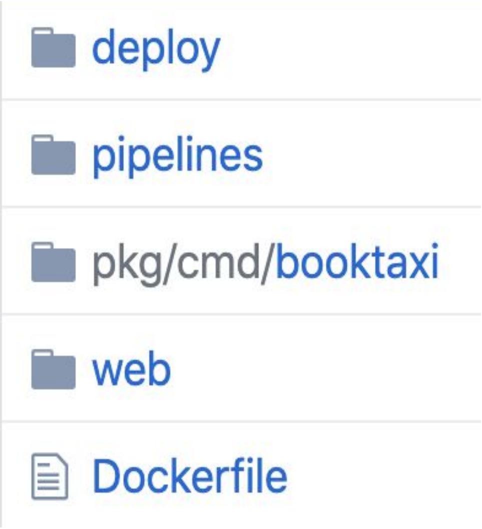
Argo CD - Challenges

It’s not all rainbows and unicorns

- Repo structure for manifests:
  - Monorepo; or
  - Separate repos for base/environments
- Managing secrets
- Order dependent deployments
- Non-declarative requirements
- Integrating with CI/CD tools (Jenkins, OpenShift Pipelines, etc)
  - Does CI/CD or Argo CD manage deployments?

Approach 1: Multiple repositories

/taxi.git



deploy

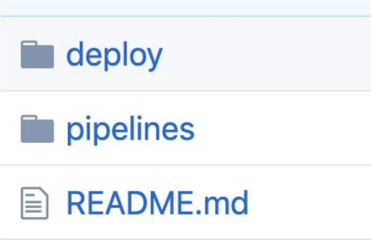
pipelines

pkg/cmd/booktaxi

web

Dockerfile

/taxi-config-**stage.git**

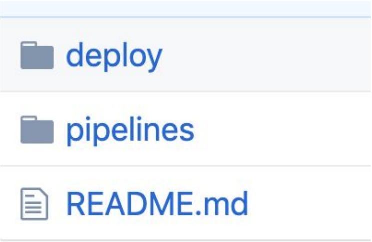


deploy

pipelines

README.md

/taxi-config-**prod.git**

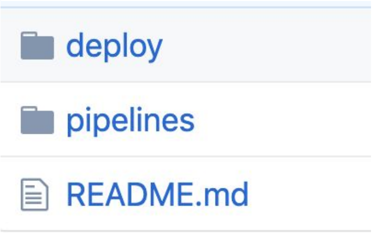


deploy

pipelines

README.md

/taxi-config-**test.git**

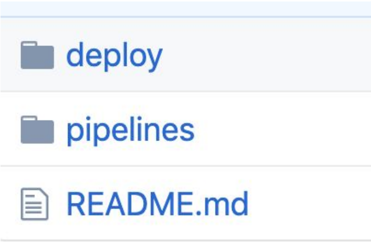


deploy

pipelines

README.md

/taxi-config-**dev.git**



deploy

pipelines

README.md

Approach 2 : Single Repository

```
├── apps
│   └── app-1
│       ├── base
│       │   └── kustomization.yaml
│       └── dev
│           ├── deployment.yaml
│           └── kustomization.yaml
├── envs
│   ├── base
│   │   ├── 205-serviceaccount.yaml
│   │   └── kustomization.yaml
│   └── dev
│       └── kustomization.yaml
└── services
    └── service-a
        ├── base
        │   ├── config
        │   │   ├── 300-deployment.yaml
        │   │   ├── 310-service.yaml
        │   │   └── kustomization.yaml
        │   └── kustomization.yaml
        └── dev
            ├── dev-deployment.yaml
            ├── dev-service.yaml
            └── kustomization.yaml
```

```
├── base
│   ├── deployment.yaml
│   ├── kustomization.yaml
│   └── service.yaml
└── overlays
    ├── development
    │   ├── kustomization.yaml
    │   └── replicas.yaml
    ├── production
    │   ├── kustomization.yaml
    │   ├── replicas.yaml
    │   └── volumes.yaml
    └── staging
        ├── kustomization.yaml
        └── volumes.yaml
```

```
├── 00-tekton
│   ├── release.notags.yaml
│   └── release.yaml
├── 01-namespaces
│   ├── cicd-environment.yaml
│   ├── dev-environment.yaml
│   └── stage-environment.yaml
├── 02-serviceaccount
│   ├── demo-sa-admin-dev.rolebinding.yaml
│   ├── demo-sa-admin-stage.rolebinding.yaml
│   ├── role-binding.yaml
│   ├── role.yaml
│   └── serviceaccount.yaml
├── 03-tasks
│   ├── buildah-task.yaml
│   ├── create-github-status-task.yaml
│   ├── deploy-from-source-task.yaml
│   └── deploy-using-kubectl-task.yaml
├── 04-templatesandbindings
│   ├── dev-cd-deploy-from-master-binding.yaml
│   ├── dev-cd-deploy-from-master-template.yaml
│   ├── dev-ci-build-from-pr-binding.yaml
│   ├── dev-ci-build-from-pr-template.yaml
│   ├── stage-cd-deploy-from-push-binding.yaml
│   ├── stage-cd-deploy-from-push-template.yaml
│   ├── stage-ci-dryrun-from-pr-binding.yaml
│   └── stage-ci-dryrun-from-pr-template.yaml
├── 05-ci
│   ├── dev-ci-pipeline.yaml
│   └── stage-ci-pipeline.yaml
├── 06-cd
│   ├── dev-cd-pipeline.yaml
│   └── stage-cd-pipeline.yaml
├── 07-eventlisteners
│   └── cicd-event-listener.yaml
└── 08-routes
    └── github-webhook-event-listener.yaml
```

Argo CD - Managing Secrets

How do I store Kubernetes secrets securely in git when a secret is not encrypted and is only base64?

- Externalize the secret using products like Vault
- Encrypt the secret in git:
  - Bitnami Sealed Secrets
  - Mozilla SOPs/KSOPs
  - Many others

Argo CD - Order Dependent Deployments

- Sometimes you have cases where you need to deploy things in a specific order
  - Subscribe Operator before deploying instance
  - Create namespace/project before deploying application into it
  - Deploy required infrastructure before application (try to avoid this)
- Tools like kustomize and helm will handle this automatically in some cases
- Argo CD provides Sync Phases and Waves to address other use cases
  - Three sync phases - Pre-sync, sync, post-sync
  - Within each phase can have multiple waves, next wave does not proceed until previous phase is healthy

https://argoproj.github.io/argo-cd/user-guide/sync-waves/

Argo CD - Non-declarative Requirements

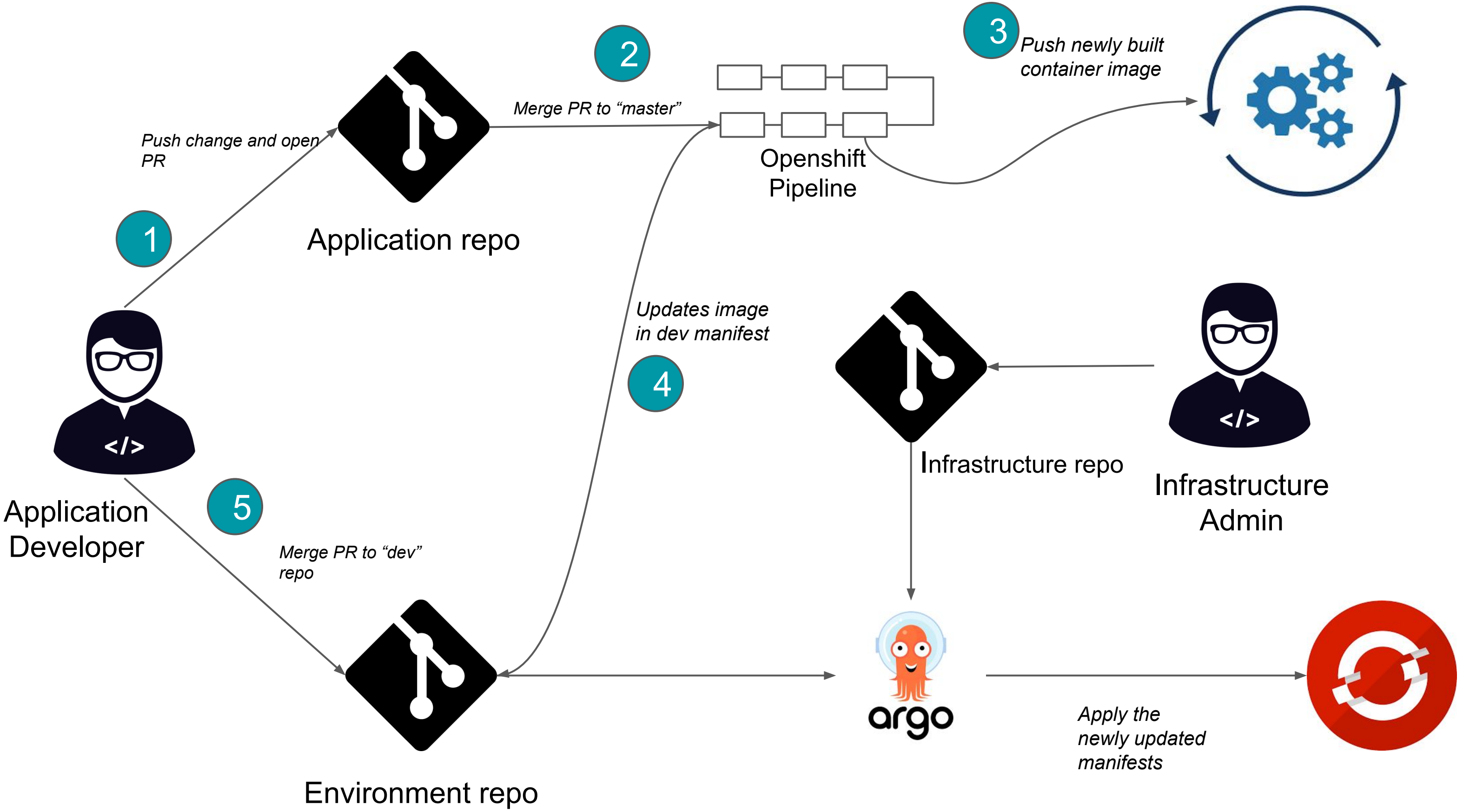
- There can be instances where you need to deploy something which cannot fully be done in a declarative way, i.e. must be scripted
- Try to minimize this and leverage kubernetes primitives where possible:
  - Init containers
  - Jobs
  - Operators
- Argo CD Resource Hooks
  - Hooks are ways to run scripts before, during, and after a Sync operation
  - Hooks can be run: PreSync, Sync, PostSync and SyncFail

Argo CD - Integrating with CI/CD Tools

The name isn’t Argo CI/CD!

CI/CD tools like Jenkins, OpenShift Pipelines still required to manage SDLC

	ArgoCD Managed Deployment	Pipeline Managed Deployment
Pro	Consistent	Post-Test update of image reference
Con	Image reference updated in git before integration tests, manage rollback?	Inconsistent
Con	Pipeline tools must be able to wait for sync	


2
3
Push newly built container image
Merge PR to "master"
Push change and open PR
Openshift Pipeline
1
Application repo
Updates image in dev manifest
4
Application Developer
Infrastructure repo
Infrastructure Admin
5
Merge PR to "dev" repo
argo
Apply the newly updated manifests
Environment repo

Red Hat

Argo CD - Avoiding Duplication

Argo CD enables deployment across multiple clusters, awesome!

Wait, how do we manage configuration without copying and pasting yaml everywhere?

Kustomize - What is it?

Kustomize lets you customize raw, template-free YAML files for multiple purposes, leaving the original YAML untouched and usable as is.

- Kustomize is a patching framework
- Enables environment specific changes to be introduced without duplicating yaml unnecessarily
- Unlike templating frameworks, all yaml can be directly applied
- Kustomize is included in kubectl and oc starting in 1.14

```
oc apply -k apps/myapp/overlays/dev
```

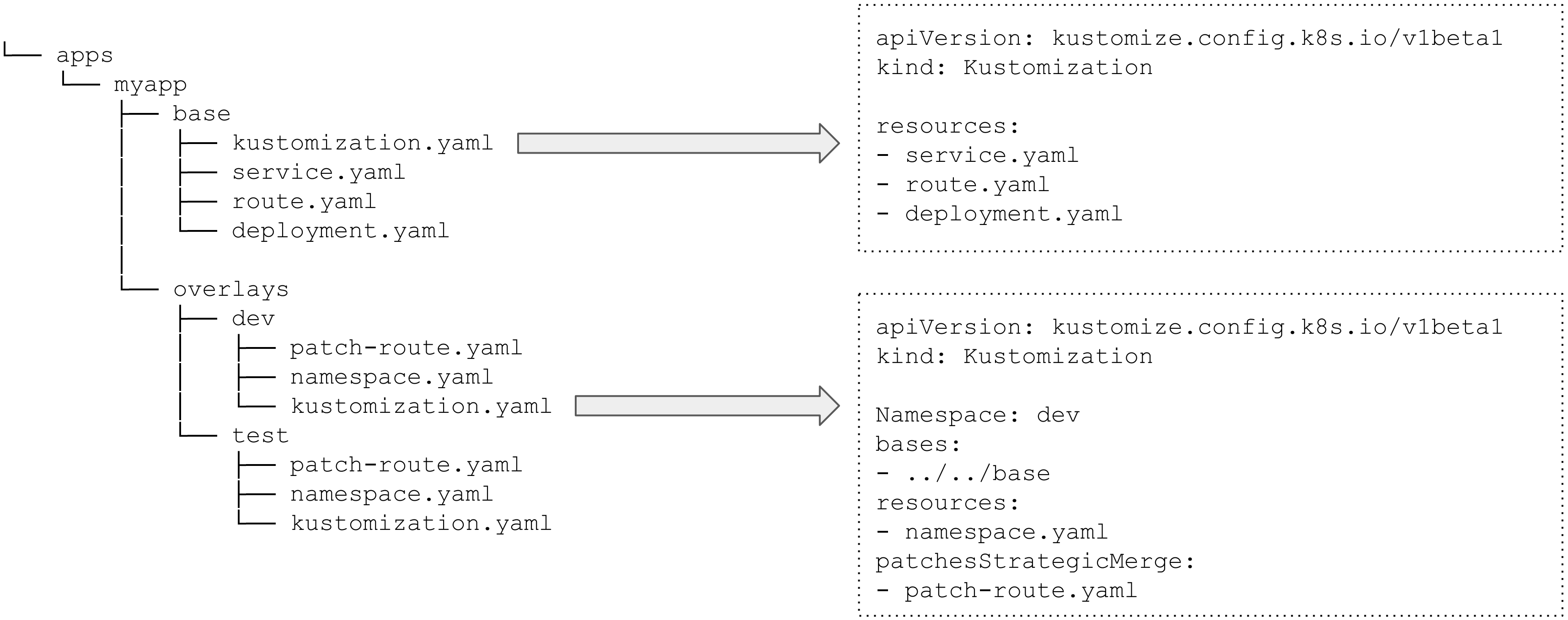
Kustomize - Organization

Kustomize is organized in a hierarchical directory structure of **bases** and **overlays**.

- A **base** is a directory with a kustomization.yaml file that contains a set of resources and associated customization.
  - A base has no knowledge of an overlay and can be used in multiple overlays.
- An **overlay** is a directory with a kustomization.yaml file that refers to other kustomization directories as its bases
  - An overlay may have multiple bases and it composes all resources from bases and may also have customization on top of them.

https://kubernetes.io/docs/tasks/manage-kubernetes-objects/kustomization/#bases-and-overlays

Using Kustomize



```
└── apps
    └── myapp
        ├── base
        │   ├── kustomization.yaml
        │   ├── service.yaml
        │   ├── route.yaml
        │   └── deployment.yaml
        │
        └── overlays
            ├── dev
            │   ├── patch-route.yaml
            │   ├── namespace.yaml
            │   └── kustomization.yaml
            └── test
                ├── patch-route.yaml
                ├── namespace.yaml
                └── kustomization.yaml
```

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
- service.yaml
- route.yaml
- deployment.yaml
```

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

Namespace: dev
bases:
- ../../base
resources:
- namespace.yaml
patchesStrategicMerge:
- patch-route.yaml
```

Why Kustomize?

- Eliminates needless duplication of yaml
- Enables reuse through customization (patching)
- Hierarchical structure provides flexibility
  - Overlays can leverage other bases and overlays
  - Overlays can reference remote repositories
- Included with kubernetes since 1.14
- Validates yaml before deployment

Kustomize vs Helm vs OpenShift Templates

Patching framework

Ability to apply yaml directly

Some use-cases may work better with templates

No support in OpenShift GUI console

Great for enterprise teams, not as good for independent applications

Kubernetes focused package manager based on templates

Templates enable prescribed customization, more difficult to customize outside those boundaries

First class support in OpenShift GUI console

Great for distributing applications across multiple organizations

OpenShift specific templating solution

Easy to understand and use

First class support in OpenShift GUI console

Templates are static, no support for dynamic scripting/variables

OPENSHIFT PIPELINES

Near Term

(3-6 months)

CORE

- **OpenShift Pipelines GA**
- Disconnected clusters (air-gapped)
- Proxy support
- Pipeline as code
- Unprivileged pipelines
- Pipeline logs in OpenShift logging stack

UX

- Console pages for all Tekton resources
- Console displays additional pipeline metadata
- Console contains Pipelines guided tour
- **Start pipeline wizard in VS Code**
- Enhanced validation in VS Code
- **Tekton Hub integration in VS Code**
- **CLI integration for Tekton Hub**
- Tekton extension for CodeReady Workspaces

ECOSYSTEM

- **Tekton Hub launch**
- Tekton community catalog in Hub
- Multi-catalog support in Hub
- Additional Tekton tasks
- Improved S2I Tekton Tasks

Mid Term

(6-9 months)

CORE

- Unprivileged pipelines
- Auto-pruning pipeline runs and task runs
- Pipeline admin metrics in Prometheus
- In-cluster Tekton catalog and hub
- Jenkins migration guide
- Deployment pattern custom tasks

UX

- Enhance pipeline builder in Console
- Expose Pipeline Dev metrics in Console
- Add Advanced pipeline templates in Console
- IntelliJ integration with Tekton Hub
- IntelliJ gains Pipeline diagram

ECOSYSTEM

- Additional official Tekton catalogs
- App Services (MW) Tasks
- Community-contributions
- **OCI artifacts for task distribution**
- Additional Tekton resource types in Hub

Long Term

(9+ months)

CORE

- Pipeline pause and resume
- Partial pipeline execution
- Notifications
- Git provider PR status integration
- Argo CD integrations

UX

- Expose Pipeline Admin metrics in Console
- Enhanced pipeline visualization in Console
- Console integration with Tekton Hub

ECOSYSTEM

- ISV Tasks in Catalog
- OCI Tekton artifact support in OpenShift
- Quality indicators in Tekton Hub

Product Manager: Siamak Sadeghianfar

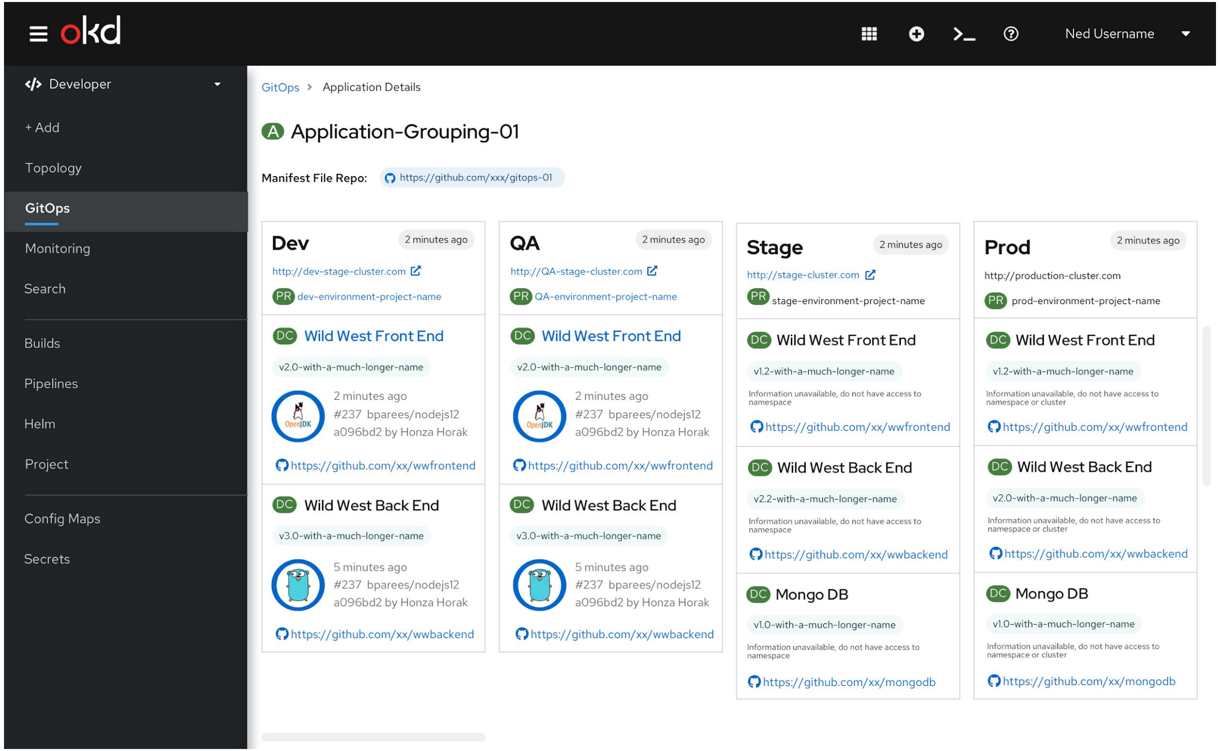
Argo CD on OpenShift

- Declarative GitOps operator for continuous delivery on Kubernetes
- Git as the single source of truth in sync with Kubernetes clusters with drift detection
- Red Hat joins Argo steering committee together with Intuit, BlackRock and Alibaba (announcement at KubeCon EU)
- Tekton and Argo CD as the basis of developer GitOps workflow

Dedicated GitOps View

Empower developers with visibility of their application across all environments

- Dedicated GitOps view
- View all app groupings
- Drill into app grouping details to get visibility into the composition and status of the applications/workloads deployed across environments
- Link out to Argo CD
- Eventually powered by Argo CD



GitOps › Application Details

A Application-Grouping-01

Manifest File Repo: https://github.com/xxx/gitops-01

Dev 2 minutes ago
http://dev-stage-cluster.com
PR dev-environment-project-name

DC Wild West Front End
v2.0-with-a-much-longer-name
2 minutes ago
#237 bparees/nodejs12
a096bd2 by Honza Horak
https://github.com/xx/wwfrontend

DC Wild West Back End
v3.0-with-a-much-longer-name
5 minutes ago
#237 bparees/nodejs12
a096bd2 by Honza Horak
https://github.com/xx/wwbackend

QA 2 minutes ago
http://QA-stage-cluster.com
PR QA-environment-project-name

DC Wild West Front End
v2.0-with-a-much-longer-name
2 minutes ago
#237 bparees/nodejs12
a096bd2 by Honza Horak
https://github.com/xx/wwfrontend

DC Wild West Back End
v3.0-with-a-much-longer-name
5 minutes ago
#237 bparees/nodejs12
a096bd2 by Honza Horak
https://github.com/xx/wwbackend

Stage 2 minutes ago
http://stage-cluster.com
PR stage-environment-project-name

DC Wild West Front End
v1.2-with-a-much-longer-name
Information unavailable, do not have access to namespace
https://github.com/xx/wwfrontend

DC Wild West Back End
v2.2-with-a-much-longer-name
Information unavailable, do not have access to namespace
https://github.com/xx/wwbackend

DC Mongo DB
v1.0-with-a-much-longer-name
Information unavailable, do not have access to namespace
https://github.com/xx/mongodb

Prod 2 minutes ago
http://production-cluster.com
PR prod-environment-project-name

DC Wild West Front End
v1.2-with-a-much-longer-name
Information unavailable, do not have access to namespace or cluster
https://github.com/xx/wwfrontend

DC Wild West Back End
v2.0-with-a-much-longer-name
Information unavailable, do not have access to namespace or cluster
https://github.com/xx/wwbackend

DC Mongo DB
v1.0-with-a-much-longer-name
Information unavailable, do not have access to namespace or cluster
https://github.com/xx/mongodb

Argo CD and Developer GitOps Workflow

	Near Term (3-6 months)	Mid Term (6-9 months)	Long Term (9+ months)
Argo CD	• Argo CD Tech Preview in OperatorHub	• Argo CD GA • Argo CD Auth integration with OpenShift • Application sets • Argo CD and Tekton integrations	• ACM Collaborations
BOOTSTRAP	• GitOps-based project bootstrapping with Tekton, Argo CD, kustomize • Dev Preview of bootstrapping with odo	• Helm support • Gitlab support • Application promotion between environments • Tech preview of bootstrapping with odo	• GA of bootstrapping with odo • Bootstrapping MW apps • Bootstrapping based on devfile
CONSOLE	• Dashboard for multi-cluster deployment environments	• Argo CD integration	

Resources

- Canada Solution Architects GitOps Repo:
  - https://github.com/redhat-canada-gitops
- GitOps Repos from Red Hatters
  - https://github.com/gnunn-gitops
  - https://github.com/pittar-gitops
  - https://github.com/PixelJonas/cluster-gitops
- OpenShift and ArgoCD Introduction Video
  - https://www.youtube.com/watch?v=xYCX2EejSMc
- Free Online OpenShift Learning
  - https://learn.openshift.com/introduction/
    - Introduction to GitOps with OpenShift
    - Multi-cluster GitOps with OpenShift

Questions?

in linkedin.com/company/red-hat

▶ youtube.com/user/RedHatVideos

f facebook.com/redhatinc

twitter.com/RedHat